NCRC-0011-US (9172)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: CONTROLLING SOFTWARE COMPONENTS IN A MULTI-NODE PROCESSING SYSTEM

INVENTORS: Hoa Thu Tran and Matthew Dickey

Express Mail No.: EL542038310US
Date: June 5, 2000

Prepared by: Trop, Pruner & Hu, P.C.
8554 Katy Freeway, Ste. 100, Houston, TX 77024
713/468-8880 [Office], 713/468-8883 [Fax]

CONTROLLING SOFTWARE COMPONENTS IN A MULTI-NODE PROCESSING SYSTEM

BACKGROUND

The invention relates to controlling software components in a multi-node processing system.

Software in a computer system may be made up of many layers. The highest layer is usually referred to as the application layer, followed by lower layers that include the operating system, device drivers (which usually are part of the operating system), and other layers. In a system that is coupled to a network, transport and network layers may also be present.

Software components may be installed and loaded as programs in a system. A program is an executable file, and when the program is executed, it is run as a process (e.g., in a UNIX system), a thread (e.g., in a WINDOWS® system), or some other execution unit. During operation of the system, various software components may be started to perform useful tasks.

Software components may be executed on different types of systems, including single processor systems, multiprocessor systems, or multi-node parallel processing systems. Examples of single processor systems include standard desktop or portable systems. A multiprocessor system may include a single node that includes plural processors running in the node. Such systems may include symmetric multiprocessor (SMP) systems. A multi-node parallel processing system may include multiple nodes that may be connected by an interconnect network. Software components may be executed on each of these nodes and run in parallel.

It is not always easy to control (e.g., start up, monitor, or terminate) processes in a multi-node parallel processing system. Because of the distributed nature of a system with multiple nodes, a mechanism may not always be available for control of the software components in the several nodes. Thus, for a parallel software application having software components that can run in a multi-node parallel processing system, a need continues to exist for a convenient and efficient method and apparatus to control software components in the multiple nodes.

SUMMARY

In general, according to one embodiment, a method of controlling software components in a processing system having plural nodes includes receiving a request to start the system and determining one or more selected software components to start in each node. Services are invoked with a manager module to start the selected software components in the nodes of the processing system.

In general, according to another embodiment, a system includes a plurality of nodes and software components in corresponding nodes. A manager module controls the software components in the plural nodes and enables a monitoring module to monitor a status of at least one of the software components.

Other features and embodiments will become apparent from the following description, from the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of an embodiment of a multi-node parallel processing system.

Fig. 2 illustrates components in nodes of the multi-node parallel processing system of Fig. 1 in accordance with an embodiment.

Fig. 3 is a flow diagram of a process to start software components in the multi-node parallel processing system in accordance with an embodiment.

Figs. 4-9 illustrate interactions between various software modules and routines in the multi-node parallel processing system to start software components in the system.

DETAILED DESCRIPTION

In the following description, numerous details are set forth to provide an understanding of the present invention. However, it will be understood by those skilled in the art that the present invention may be practiced without these details and that numerous variations or modifications from the described embodiments may be possible. For example, although reference is made to use of WINDOWS® operating systems in some described embodiments, other embodiments may employ other types of operating systems.

Referring to Fig. 1, a multi-node parallel processing system 10 includes plural nodes (12A, 12B) coupled by an interconnect network 14. Although two nodes are shown in Fig. 1, additional nodes may be coupled by the interconnect network 14. Each of the nodes (12A, 12B) may be referred to as a "node 12." In the example embodiment of Fig. 1, the node 12A is a master node while the node 12B is a slave node. Multiple slave nodes may be coupled to the interconnect network 14.

The interconnect network 14 may be any of various types of networks. For example, the interconnect network 14 may be a local area network (LAN), a wide area network (WAN), or another type of network. Each of the nodes 12 includes a network interface to enable communication over the interconnect network 14.

The nodes 12A and 12B include respective operating systems 18 and 20. In one embodiment, the operating systems (18, 20) are WINDOWS® NT operating systems, WINDOWS® 2000 operating systems, or other WINDOWS® operating systems from Microsoft Corporation. In further embodiments, other types of operating systems may be used. Various software components may be executable in each of the nodes 12 in the system 10. Some of these components may be implemented as services, such as WINDOWS® NT services, which are background processes that are loaded at boot time or during operation of the system. A WINDOWS® NT service is a program that runs in a loop waiting for input, such as user requests from a control program or system events such as messages or process terminations.

In a WINDOWS® NT environment, a service is loaded and controlled by a service controller manager (SCM). Fig. 1 shows an SCM 22 in node 12A and an SCM 24 in node 12B. Services that may be executable in each node 12 include data servers, which are database management modules that control access to storage devices in each node 12.

In the master node 12A, an additional service is a query coordinator 32, which manages the communication of queries to the appropriate one of the data servers 26, 28, and 30 in response to receipt of a database query from a client. In further embodiments, other types of software applications, implemented as services, may be executable in the parallel processing system 10. As used here, a "service" refers to a process or other execution unit that, while running, waits for predetermined inputs or events in the system to perform predefined tasks.

A database managed by the data servers and query coordinator may be an object relational database distributed among the plural nodes 12. In one example, the database is part of a data warehouse system that stores, manages, analyzes, and reports transaction data, e.g., retail transaction data. Generally, a data warehouse may include copies of transaction data that have been structured for query and analysis or reporting. For example, retail transaction data may track purchases made by users of goods and services from participating retail sources. The data warehouse system performs the collection, management, analysis, and reporting of such transaction data. The volume of data involved in a data warehouse system may be extremely large, typically in the gigabyte range and sometimes in the terabyte range.

A start procedure 34 may be invoked in the master node 12A to control startup of desired software components in each of the nodes 12. Through an application programming interface (API), the start procedure 34 cooperates with the SCM 22 and SCM 24 to control the startup of the query coordinator 32, the data servers 26, 28, and 30, and any other software component in the system 10.

In addition, other utilities may be employed to perform other control tasks in connection with the query coordinator 32 and data servers 26, 28, and 30. For example, such other control tasks may include terminating or shutting down various software components, monitoring the status of the software components, and changing configurations of the software components.

Referring to Fig. 2, further components of the master node 12A and slave node 12B are illustrated. In the master node 12A, a user interface 100 is provided through which the user can issue requests to perform various control tasks with respect to software components in the system 10. The user interface 100 may include a control panel, a utility, or other user interfaces (whether graphical or text). The start procedure 34 includes a Start service 102 and a Start.exe program 104. The Start service 102 is responsible for handling the WINDOWS® NT services protocol (including communication with the SCM 22) and for initiating the Start.exe program 104. The start service 102 also monitors the status of the process it spawns, in this case the Start.exe process 104. As used here, when a program is executed, it is referred to as a process,

which may refer to any execution unit, such as a thread in the WINDOWS® context or a process in the UNIX context.

The Start.exe process 104 acts as a service control program for the query coordinator 32 and the data servers 26, 28, and 30. The Start.exe process 104 starts the query coordinator 32 and data servers 26, 28, and 30 by launching, through the SCMs 22 and 24, their corresponding server services 106, 108, 110, and 112 (in the illustrated system 10). The server service 106 is responsible for starting up the query coordinator 32, while the server services 108, 110, and 112 are responsible for starting up respective data servers 26, 28, and 30. In addition, each of the server services (106, 108, 110, and 112) is responsible for handling the WINDOWS® NT services protocol on behalf of a corresponding child process and for monitoring the child process (one of the query coordinator and data servers).

Once a child process (e.g., the query coordinator 32 or one of the data servers 26, 28, and 30) is started, the server service 106, 108, 110 or 112 monitors the child process and waits for the termination of the corresponding child process. When the child process terminates, the server service 106, 108, 110 and 112 reports the exit code to the SCM (22 or 24) and kills itself.

Instructions associated with the various software applications, modules, routines, or services in the master node 12A may be executable on one or more control units 120. The instructions, and data associated with the instructions, may be stored in one or more storage units 122. A network interface 124 is provided to enable communication between the master node 12A and the interconnect network 14.

Similarly, the instructions of the various software applications, modules, routines, or services in the slave node 12B are executable on one or more control units 130. Instructions and data may be stored in one or more storage units 132 in the node 12B. A network interface 134 enables communication between the interconnect network 14 and the slave node 12B. The network interfaces 124 and 134 in respective nodes 12A and 12B may include network controller cards or chips. The network interfaces 124 and 134 may also include appropriate device drivers and transport and network stacks. As examples, the transport and network stacks may include TCP/IP and/or UDP/IP stacks. IP stands for Internet Protocol, with versions described in Request for Comments (RFC)

791, entitled "Internet Protocol," dated September 1981; and RFC 2460, entitled "Internet Protocol, Version 6 (IPv6) Specification," dated December 1998. TCP is described in RFC 793, entitled "Transmission Control Protocol," dated September 1981; and UDP is described in RFC 768, entitled "User Datagram Protocol," dated August 1980. TCP and UDP are transport layers for managing connections between network elements over an IP network.

Generally, in one aspect of some embodiments of the invention, a coordinated method and apparatus is provided to start software components in plural nodes in a parallel processing system. Instances of manager modules are started in each node, with a start procedure started in a master node. The start procedure issues requests to the manager module instances in each of the nodes to start respective services that are responsible for spawning the desired software components. In a WINDOWS® NT environment, the manager module is a service control manager, with an instance of the service control manager run on each of the nodes in the parallel processing system.

From a single node, desired software components may be started or otherwise controlled in a multi-node system. In addition, by using separate services (such as the starter service 102 and server services 106, 108, 110, and 112) to handle the WINDOWS® NT services protocol, ease of software development is enhanced.

Referring to Fig. 3, a technique for starting the query coordinator 32 and data servers 26, 28, and 30 in the parallel processing system 10 is illustrated. A similar technique may be applied to other software components in the system 10. In response to a command to start the software components, which may be entered by a user or by some other event, the master node 12A launches (at 202) the Start service 102. The Start service 102 then registers (at 203) with the SCM 22. Next, the Start service 102 invokes (at 204) the Start.exe program 104. Once started and executing, the Start.exe program 104 is referred to as the Start.exe process 104. Once it launches the Start.exe process 104, the Start service 102 enters (at 205) its monitoring loop to monitor the Start.exe process 104.

The Start.exe process 104 determines (at 206) the details of the configuration of the system 10. The details may include the number of nodes and processes to be started in each node. The Start.exe process 104 then sends a request to the SCMs 22 and 24 to

launch (at 208) the server service for each of the desired child processes in the nodes 12 of the system 10. Parameters are passed with the requests issued by the Start.exe process 104. In response, each server service registers (at 209) with the SCMs 22 and 24. Each server service then spawns (at 210) a corresponding child process (a query coordinator or data server). The server service then enters (at 212) its monitoring loop, in which it monitors the status of the child processes, which includes the termination of the child process.

Referring to Fig. 4, the installation and configuration of services with the SCM (including the SCM 22 and SCM 24) are illustrated. A configuration module 302 accesses configuration information files 304 that may include information pertaining to the nodes of the parallel processing system 10 as well as the software components, such as the Start service 102, the server services, the data servers, and the query coordinator that are executable in the nodes 12. In installing the services, corresponding entries in an SCM database 306 are created. A first entry 308 may contain information pertaining to the Start service 102. A second entry 310 may contain information pertaining to the query coordinator 32 and its associated server service, and a third entry 312 may contain information pertaining to the data server 26 and its associated server service in the master node 12A.

Services may also be installed and configured on remote nodes, including the slave node 12B. The SCM 24 in the slave node 12B includes an SCM database 320 including a first entry 322 containing information relating to the data server 28 (and its associated server service and a second entry 324 containing information relating to the data server 30 (and its associated server service).

Referring to Fig. 5, in response to a command to start the database application, the entry 308 in the SCM database 306 is used by the SCM 22 to initiate the Start service 102. The Start service 102 includes initiation instructions 402 for starting the Start.exe process 104. The Start service 102 also includes instructions 404 to enter into a monitoring loop after the Start service 102 has spawned the Start.exe process 104.

As further shown in Fig. 6, the Starter.exe process 104 includes instructions 410 to retrieve configuration data (relating to the nodes 12, data servers, and query coordinator) from the configuration files 304. The Start.exe process 104 also includes

instructions 412 to start a child process (query coordinator or data server) in the nodes 12 of the parallel processing system 10 as well as instructions 414 to monitor the spawned child process. In the monitoring loop, the Start service 102 waits for the Starter.exe process 104 to terminate. The Starter service 102 may also accept status queries and shutdown commands.

Referring to Fig. 7, under control of the instructions 412, the Start.exe process 104 sends a request to the SCM 22 to start the query coordinator 32. In response, the SCM 22 uses the entry 310 of the SCM database 306 to initiate the first server service 106. The server service 106 contains instructions 420 to spawn the query coordinator 32. The server service 106 also contains instructions 422 for monitoring the spawned query coordinator 32.

Referring to Fig. 8, the Starter.exe process 104 issues a request to the SCM 22 to start the data server 26. Using information in the entry 312 of the SCM database 306, the SCM 22 invokes the server service 108, which contains instructions 430 to spawn the data server 26. As with the server service 106 for the query coordinator 32, the server service 108 also includes instructions 432 to monitor the status of the data server 26. Spawning of the query coordinator and data servers is accomplished in one embodiment using a WINDOWS® NT CreateProcess() call.

Referring to Fig. 9, the Starter.exe process 104 starts the data server 28 in the remote node 12B. The Starter.exe process 104 issues a request to the SCM 22, which forwards the request to the SCM 24 in the node 12B. Using information in the entry 322 of the SCM database 320, the SCM 24 invokes the server service 110 in the node 12B. Under control of instructions 440, the server service 110 spawns the data server 28. The server service 110 also includes instructions 442 to monitor the status of the data server 28.

Once the query coordinator 32 and all desired data servers 26, 28, and 30 have been started, the Starter.exe process 104 may enter its monitoring loop to check the start of server services 106, 108, 110, and 112 using status queries. Any failed data servers detected by the Starter.exe process 104 may be restarted by restarting the associated server service. If the failed component is the query coordinator 32, then the Starter.exe process 104 causes all server services to exit.

While the invention has been disclosed with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of the invention.